Angular Advanced
01 - Introduction

Peter Kassenaar
info@kassenaar.com

Peter Kassenaar

- Trainer, author, developer – since 1996
- Specialty: *"Everything JavaScript"*
- JavaScript, ES6, Angular, NodeJS, TypeScript, jQuery, Vue.js, React

www.kassenaar.com

info@kassenaar.com

Twitter: @PeterKassenaar

Angulartraining.nl
Home
Training
Dates
Information
Contact
2018 dates now available!
World-class Angular training in Dutch and English
Live classrooms - focused on today's developers
LEARN MORE
SIGN UP!

www.angulartraining.nl

About you…

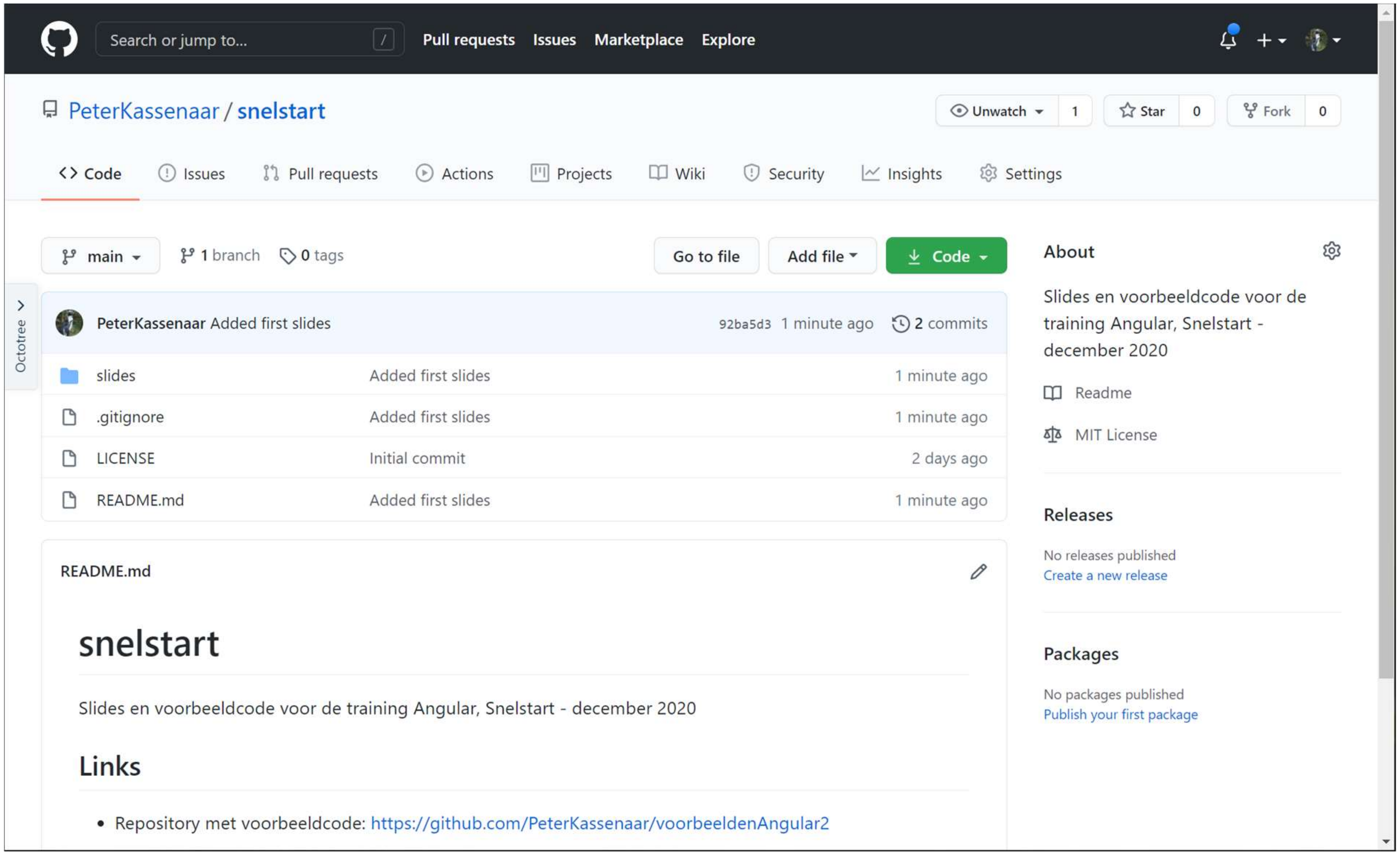
Knowledge of Angular, (mobile/web-) apps?

How long have you worked with Angular yet?

Tell us a little bit about your projects.

What are your expectations of this course?

github.com/PeterKassenaar/snelstart



PeterKassenaar Added first slides 92ba5d3 1 minute ago 2 commits

slides	Added first slides	1 minute ago
.gitignore	Added first slides	1 minute ago
LICENSE	Initial commit	2 days ago
README.md	Added first slides	1 minute ago

README.md

snelstart

Slides en voorbeeldcode voor de training Angular, Snelstart - december 2020

Links

- Repository met voorbeeldcode: https://github.com/PeterKassenaar/voorbeeldenAngular2

About

Slides en voorbeeldcode voor de training Angular, Snelstart - december 2020

Readme

MIT License

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Agenda

7-8-9 Dec. 2020 – Mo + Tu + Wed.

~9:00 start

~ 10:30 Break

~12:00 lunch

~ 14:30 Break

~16:15-16:30 closing

Wednesday – probably wrap-up a little bit early

Material

Software	(Angular + Editor + Browser + libraries)
Handouts	(PDF, Github)
Workshops	(in the presentations)
Websites	(online)

angular.io/

Short recap

- Last week: Fundamentals
  - Concepten, context & architecture
  - Angular CLI
  - Components, Data binding
  - Services
  - Live API's
  - Component communication / event buses
  - Routing
  - Forms

Broadening?

or…

deepening?

Agenda - 3 days - Thematic

- Day 1: Architecture
  - Angular CLI
  - Composing Applications with multiple modules
  - Routing and lazy loading modules
  - Loading Strategies
  - Advanced components / composition
- Day 2: Store & Observables
  - Introduction - @ngrx/store
  - Concepts, State, Action, Reducer, Dispatcher, Effect
  - More on observables…

Agenda - 3 days- Thematic

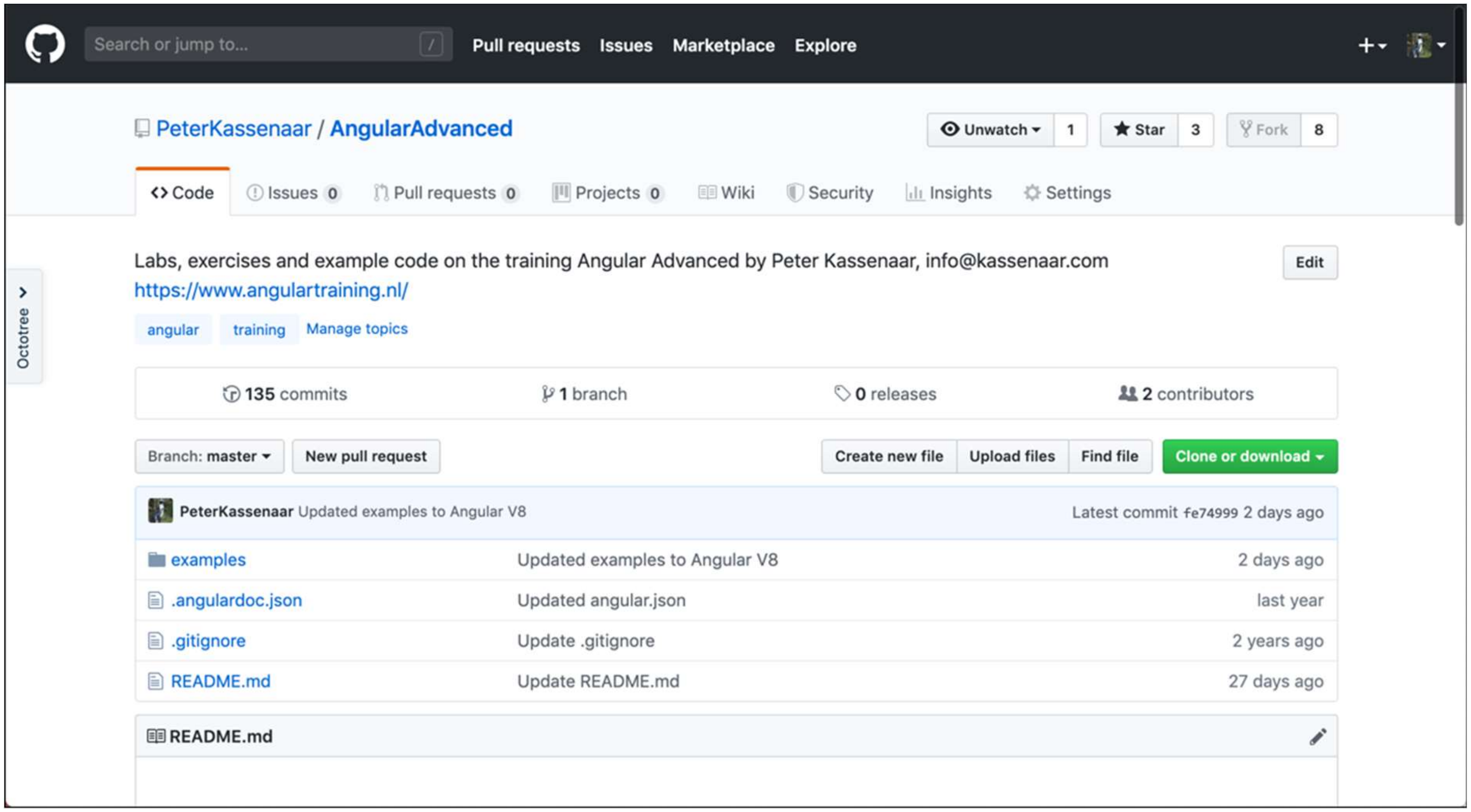
- Day 3: Miscellaneous
  - Unit Testing
  - Angular monorepo's / micro frontends?
  - Publishing Angular libraries to NPM
  - More on Angular Schematics
  - Your turn : Q & A, specific issues
  - …
- Overall : Best practices on coding & architecture

Labs and example code

1. Labs/Exercises
    - In the PDF’s in the Github-repo. But: feel free to deviate. Adapt to suit your own needs! (hobby, work, current projects)

2. Example code
    - Executions of the exercises, small projects (`npm install`, `npm start`)
    - Work in progress – let me know of additions/errors!
    - github.com/PeterKassenaar/AngularAdvanced

Generic ‘Advanced’ Github repo



Labs, exercises and example code on the training Angular Advanced by Peter Kassenaar, info@kassenaar.com
https://www.angulartraining.nl/

PeterKassenaar Updated examples to Angular V8 — Latest commit fe74999 2 days ago

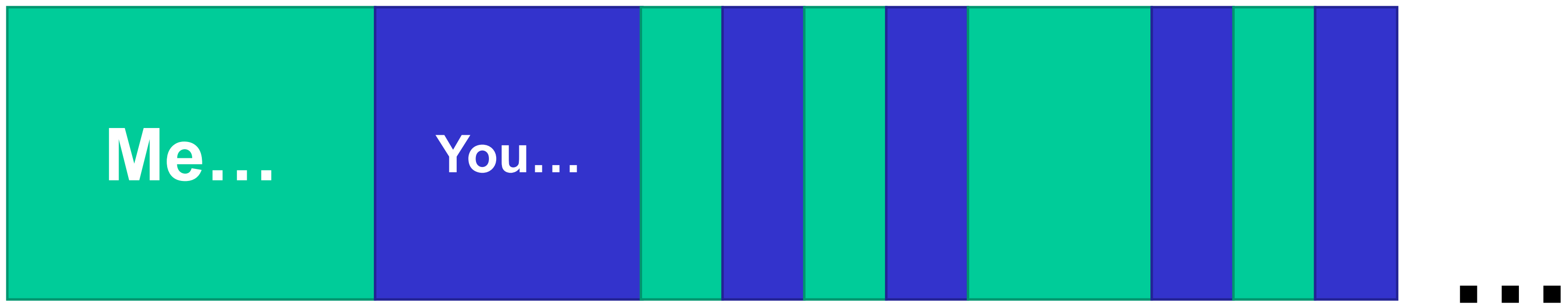
examples	Updated examples to Angular V8	2 days ago
.angulardoc.json	Updated angular.json	last year
.gitignore	Update .gitignore	2 years ago
README.md	Update README.md	27 days ago

README.md

https://github.com/PeterKassenaar/AngularAdvanced

PK1

Overall process



Slide 24

PK1	Peter Kassenaar; 31-3-2020

PK2

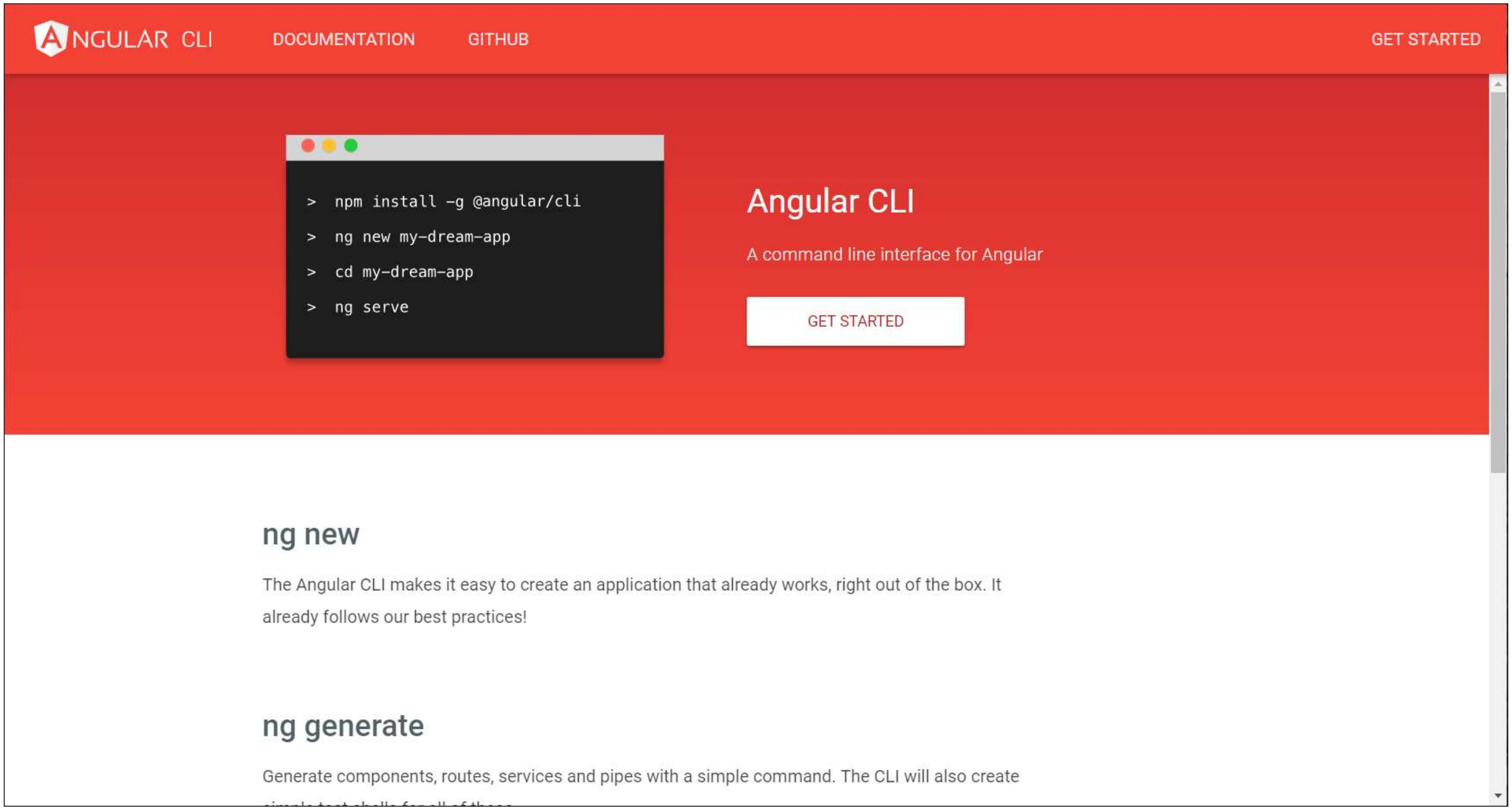
Questions?

Slide 25

PK2 Peter Kassenaar; 31-3-2020

Angular CLI

Scaffold new projects, modules, components via command line…



```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Angular CLI

A command line interface for Angular

ng new

The Angular CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

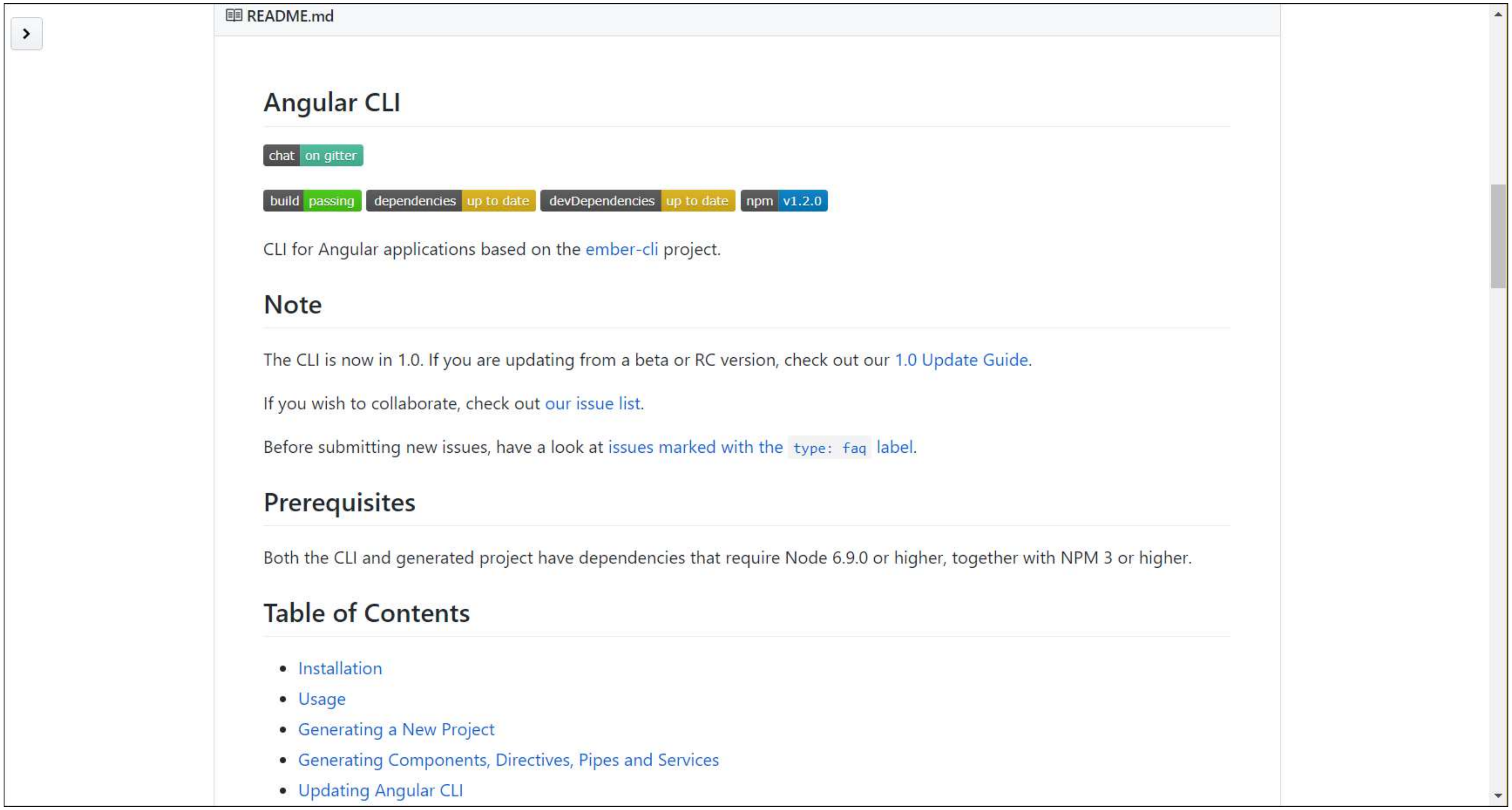
ng generate

Generate components, routes, services and pipes with a simple command. The CLI will also create

We’ll be using Angular-CLI this course

- It *is* possible to configure your Angular app by hand
- Using the CLI it’s much simpler.
- CLI-options:
    - Scaffolding
    - Generating
    - Testing
    - Building
    - AOT-Compiling
    - …

https://cli.angular.io



README.md

Angular CLI

chat on gitter

build passing | dependencies up to date | devDependencies up to date | npm v1.2.0

CLI for Angular applications based on the ember-cli project.

Note

The CLI is now in 1.0. If you are updating from a beta or RC version, check out our 1.0 Update Guide.

If you wish to collaborate, check out our issue list.

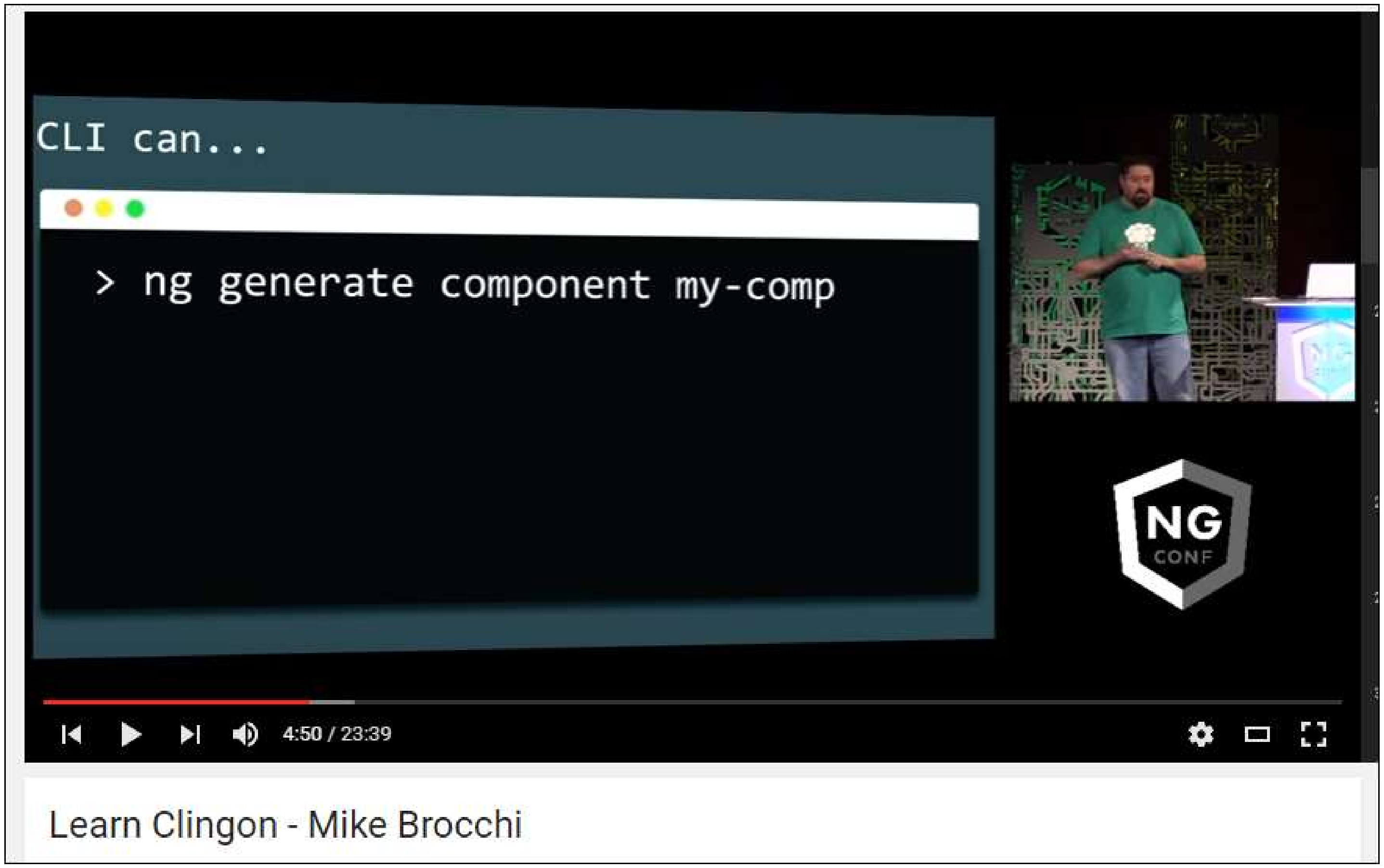
Before submitting new issues, have a look at issues marked with the `type: faq` label.

Prerequisites

Both the CLI and generated project have dependencies that require Node 6.9.0 or higher, together with NPM 3 or higher.

Table of Contents

```
npm install -g @angular/cli
```



CLI can...

```
> ng generate component my-comp
```

Learn Clingon - Mike Brocchi

https://www.youtube.com/watch?v=wHZe6gGI5RY

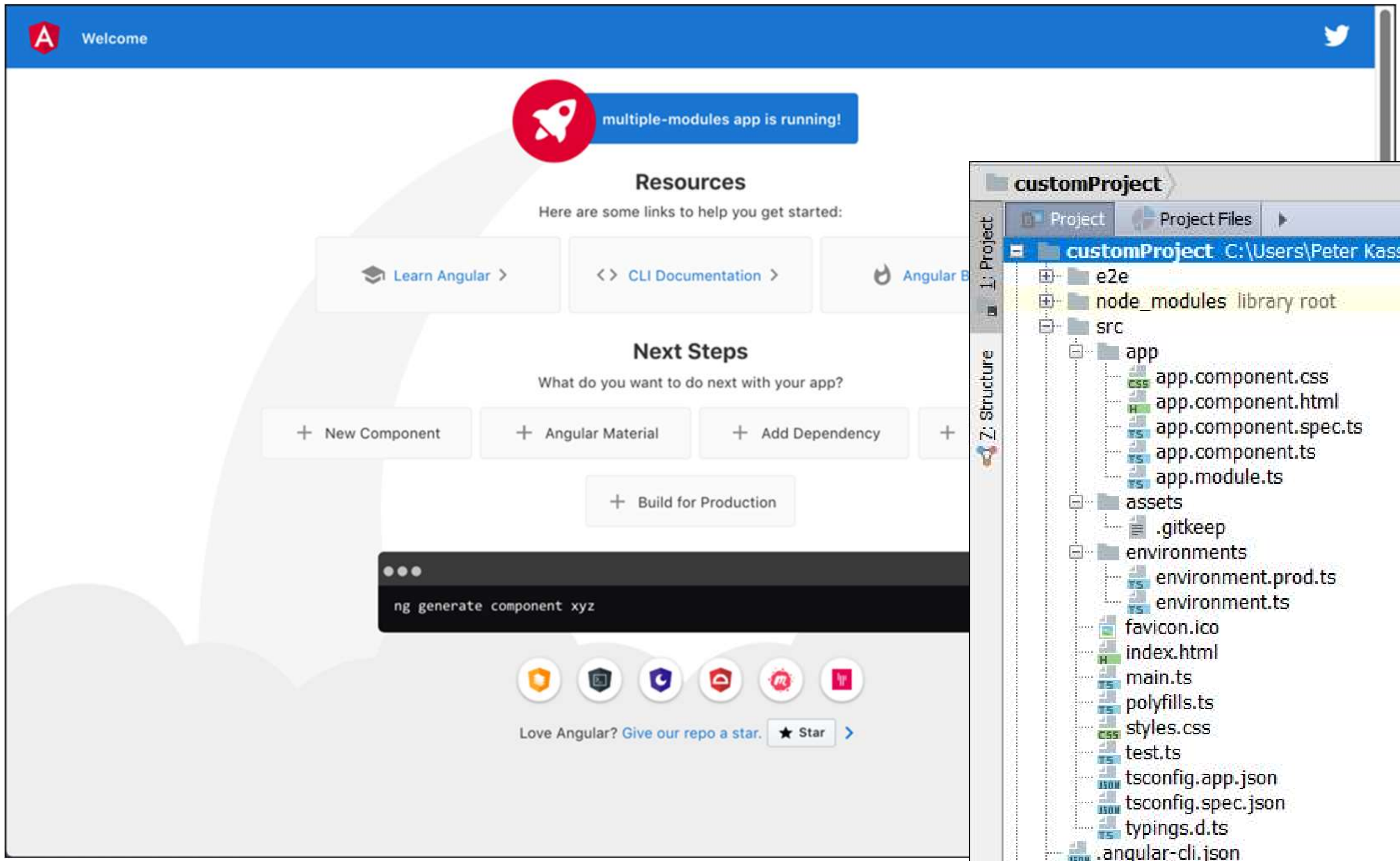
Main commands

ng new – create basic app

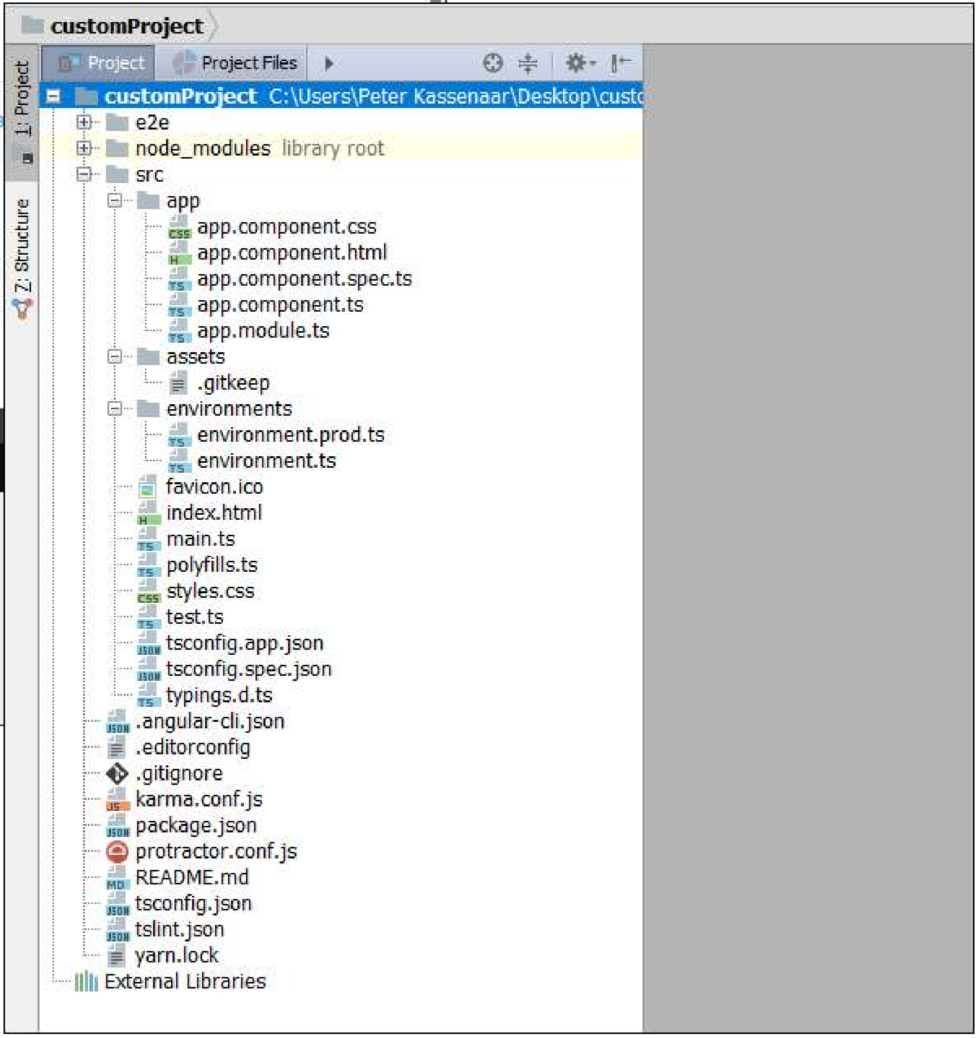
```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve
```

Project is served on `http://localhost:4200`

Default application


Welcome
multiple-modules app is running!
Resources
Here are some links to help you get started:
Learn Angular
CLI Documentation
Angular B
Next Steps
What do you want to do next with your app?
New Component
Angular Material
Add Dependency
Build for Production
Love Angular? Give our repo a star.
Star

```
ng generate component xyz
```



```
customProject C:\Users\Peter Kassenaar\Desktop\custo
  e2e
  node_modules library root
  src
    app
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
      .gitkeep
    environments
      environment.prod.ts
      environment.ts
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
    typings.d.ts
  .angular-cli.json
  .editorconfig
  .gitignore
  karma.conf.js
  package.json
  protractor.conf.js
  README.md
  tsconfig.json
  tslint.json
  yarn.lock
External Libraries
```

(228 MB)

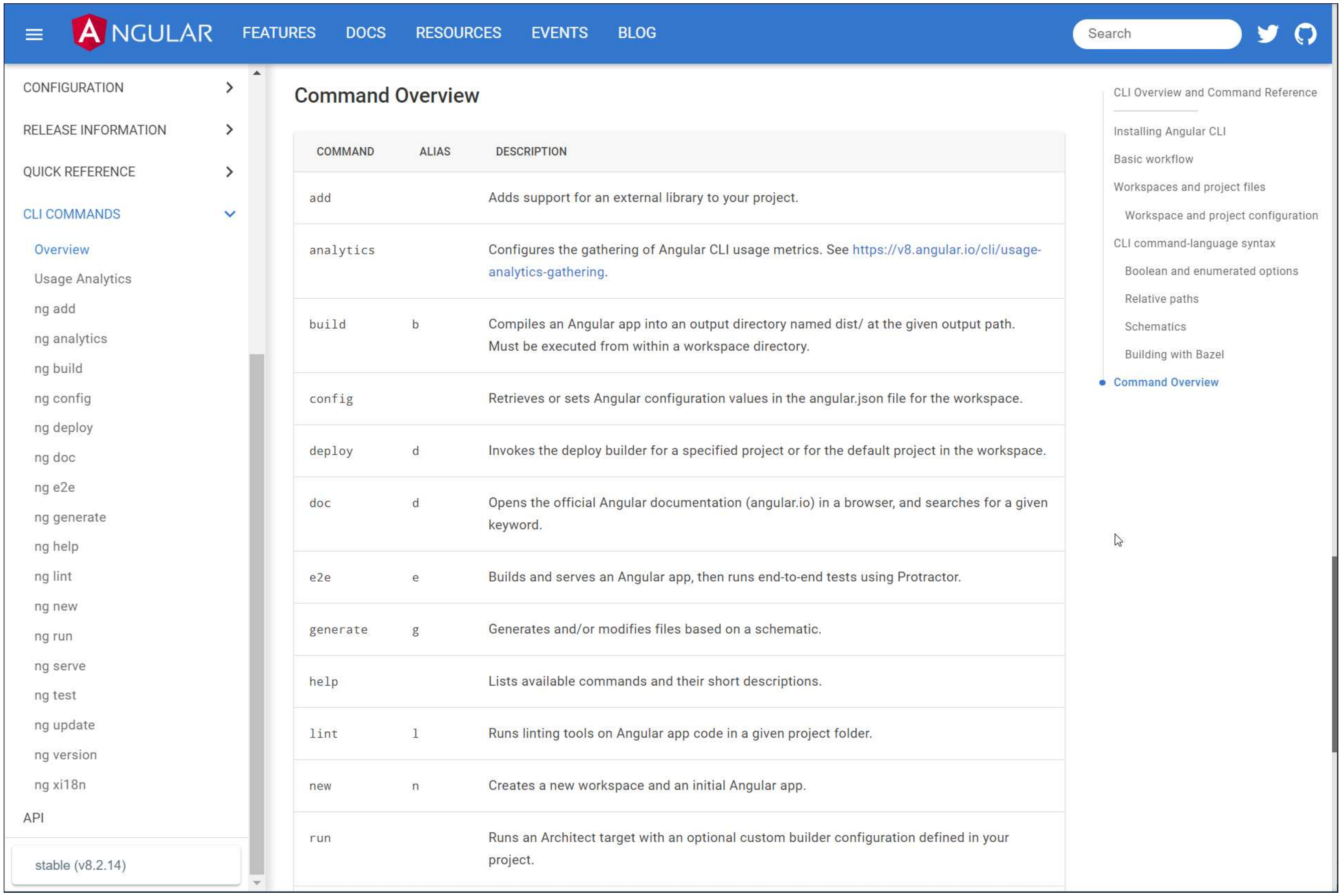
Some CLI tips & tricks

- `ng serve --open` Directly open the compiled project in the browser
- `ng serve --port 4300` Serve project on different port
- `ng serve --ssl` Serve using `https://`
- `ng serve --live-reload false` Do not use live reload
- `ng serve --help` Overview of all other options

More `ng` tooling

- `ng generate <blueprint> --dry-run` Do not write output files
- `ng generate <blueprint> --spec false` Do not write spec file
- `ng generate module <name> --routing` add routing to new module

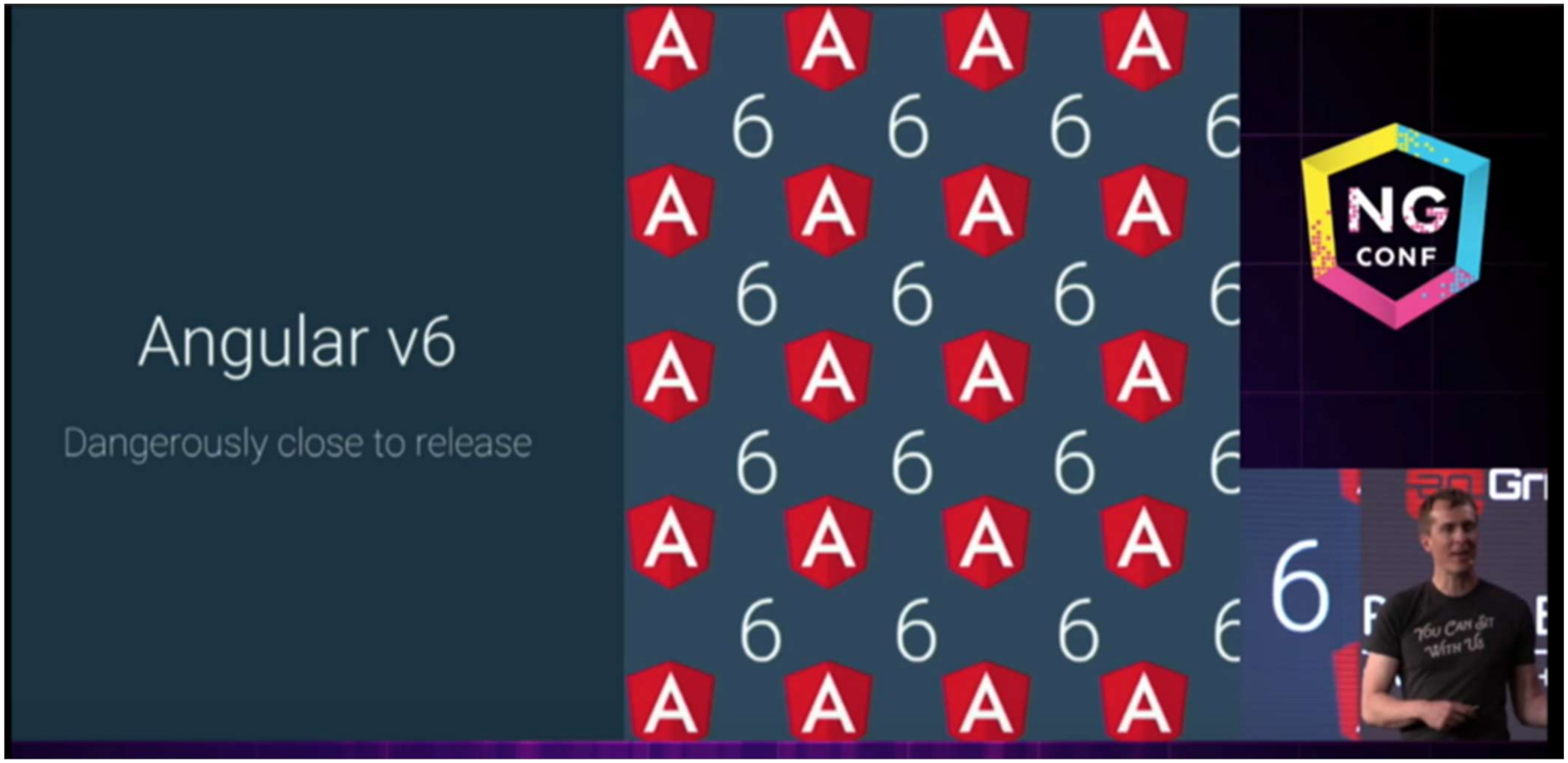
Lots (!) of options



Command Overview

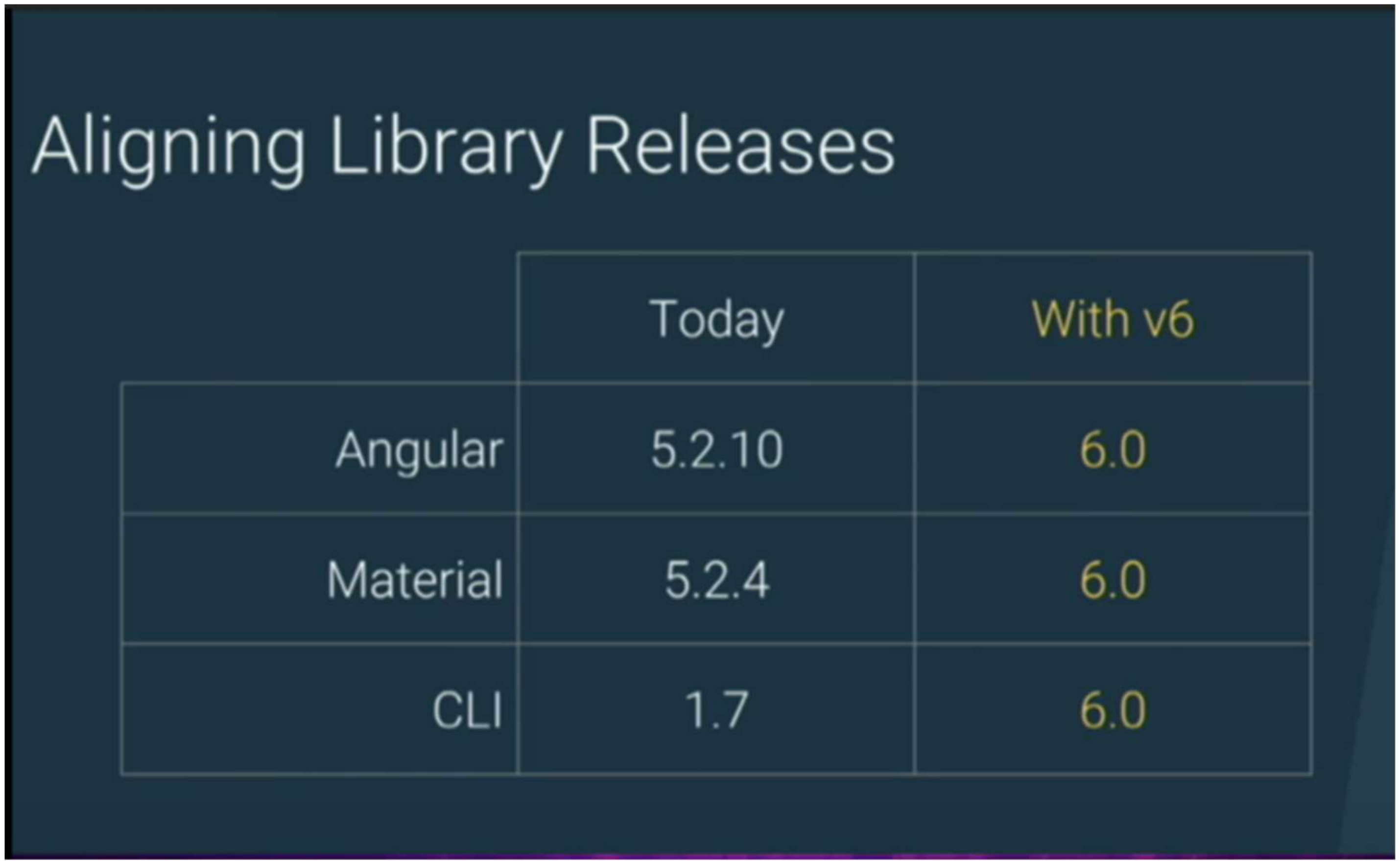
COMMAND	ALIAS	DESCRIPTION
`add`		Adds support for an external library to your project.
`analytics`		Configures the gathering of Angular CLI usage metrics. See https://v8.angular.io/cli/usage-analytics-gathering.
`build`	`b`	Compiles an Angular app into an output directory named dist/ at the given output path. Must be executed from within a workspace directory.
`config`		Retrieves or sets Angular configuration values in the angular.json file for the workspace.
`deploy`	`d`	Invokes the deploy builder for a specified project or for the default project in the workspace.
`doc`	`d`	Opens the official Angular documentation (angular.io) in a browser, and searches for a given keyword.
`e2e`	`e`	Builds and serves an Angular app, then runs end-to-end tests using Protractor.
`generate`	`g`	Generates and/or modifies files based on a schematic.
`help`		Lists available commands and their short descriptions.
`lint`	`l`	Runs linting tools on Angular app code in a given project folder.
`new`	`n`	Creates a new workspace and an initial Angular app.
`run`		Runs an Architect target with an optional custom builder configuration defined in your project.

Angular CLI 6.0 - As of May 2018



https://www.youtube.com/watch?v=dIxknqPOWms

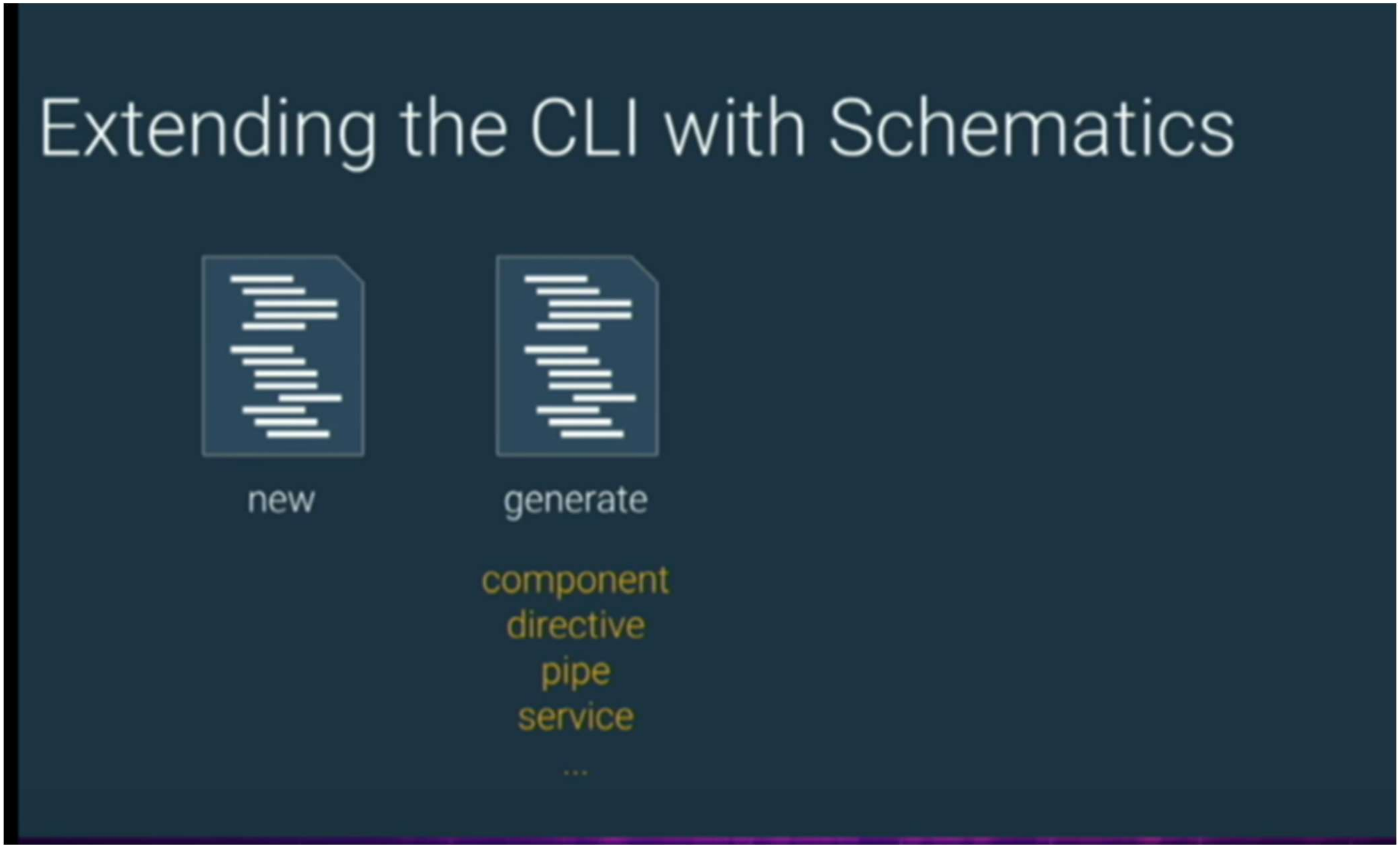
About version numbering



Aligning Library Releases

	Today	With v6
Angular	5.2.10	6.0
Material	5.2.4	6.0
CLI	1.7	6.0

New CLI Options

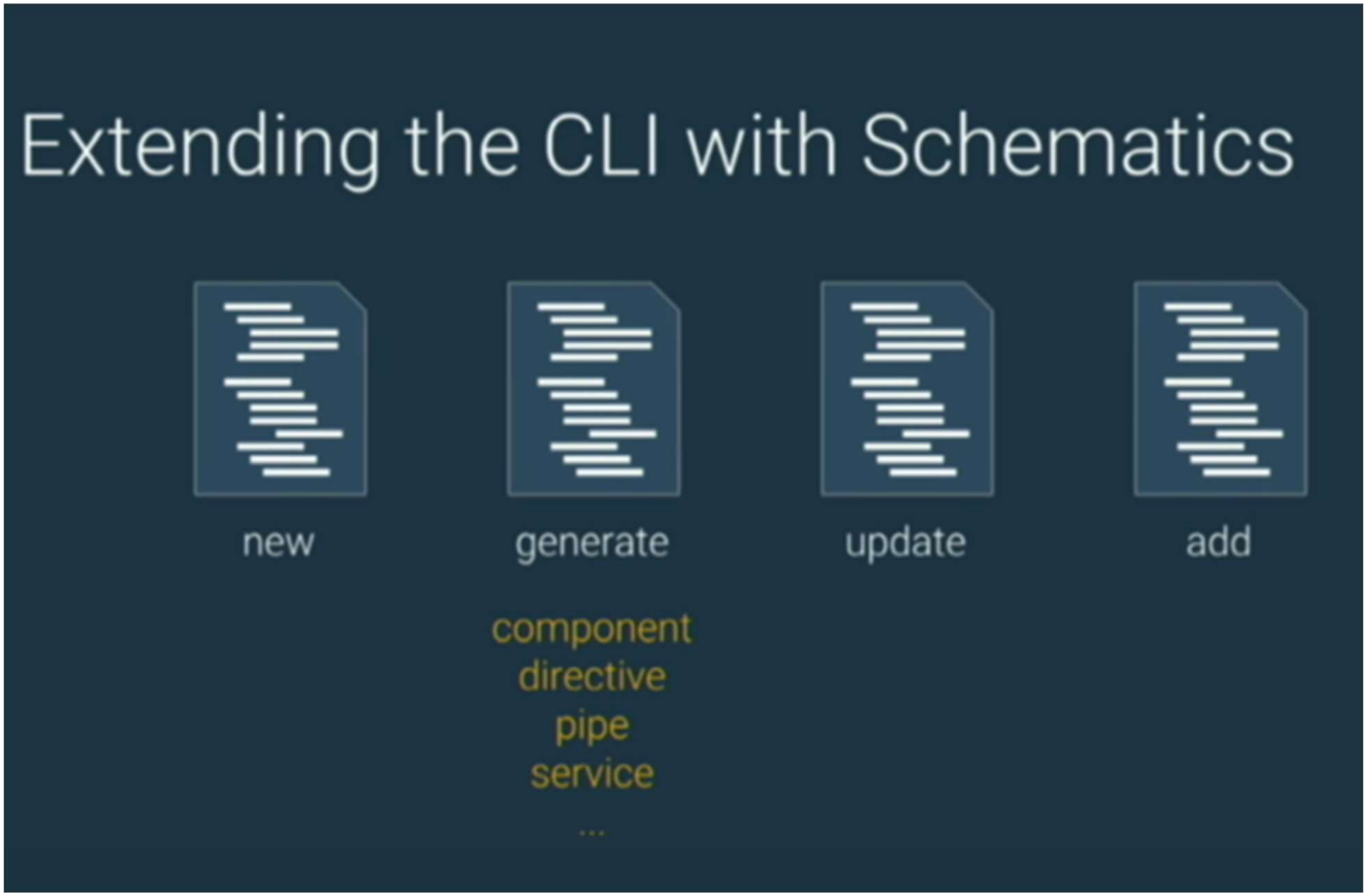


Extending the CLI with Schematics

new

generate

component
directive
pipe
service
...



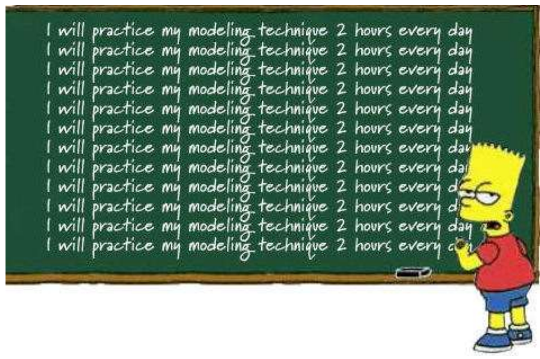
Extending the CLI with Schematics

Info on the Angular 6.x keynote

https://www.youtube.com/watch?v=dIxknqPOWms

Mini Workshop

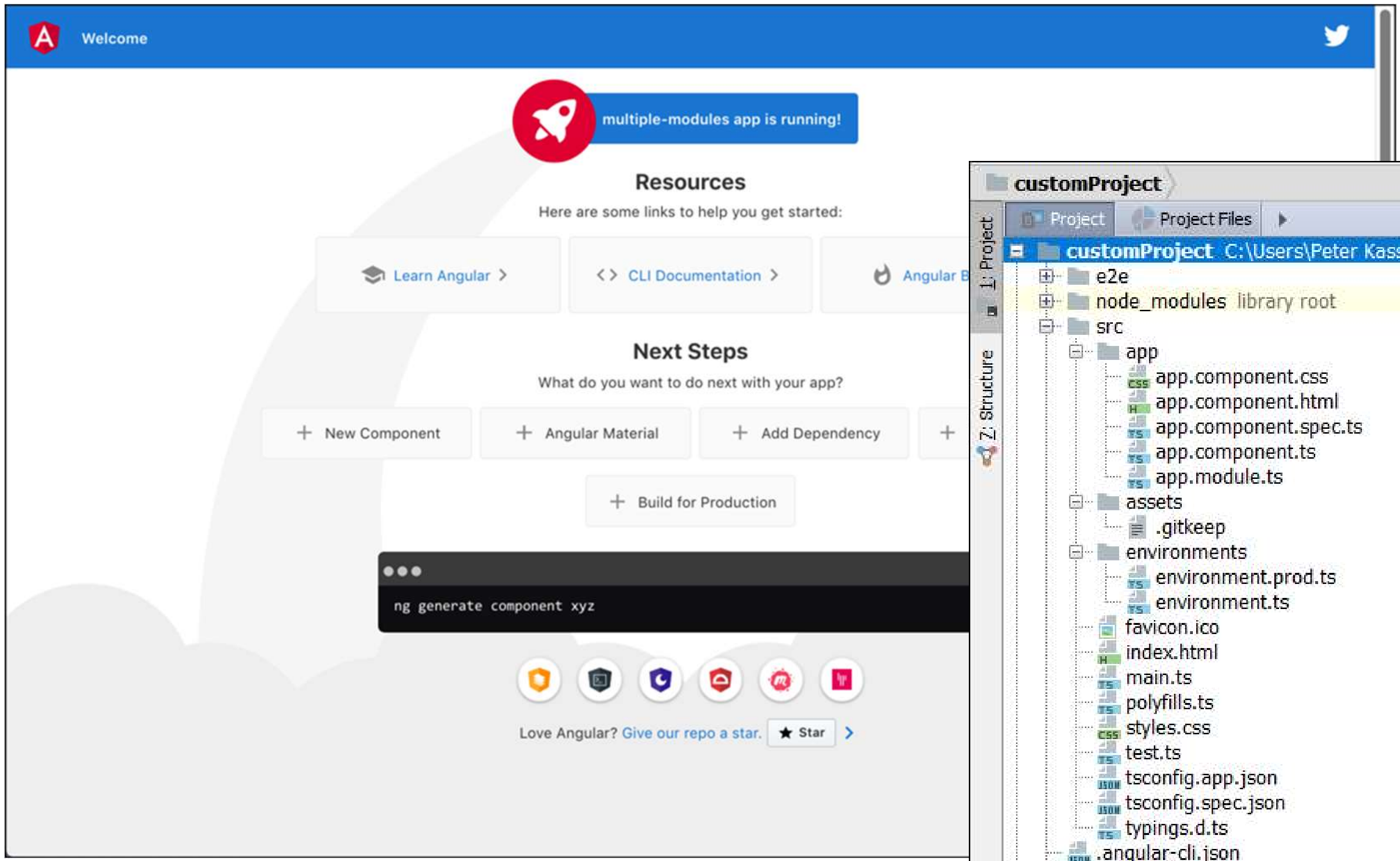
- Generate a new, blank project with Angular CLI
- Generate a new component or a new service with it
- Add some new CLI extensions, see **ngadd.com**; for instance
    - `@angular/material`
    - `@angular/elements`
    - See how/where they are installed
    - What files are affected?



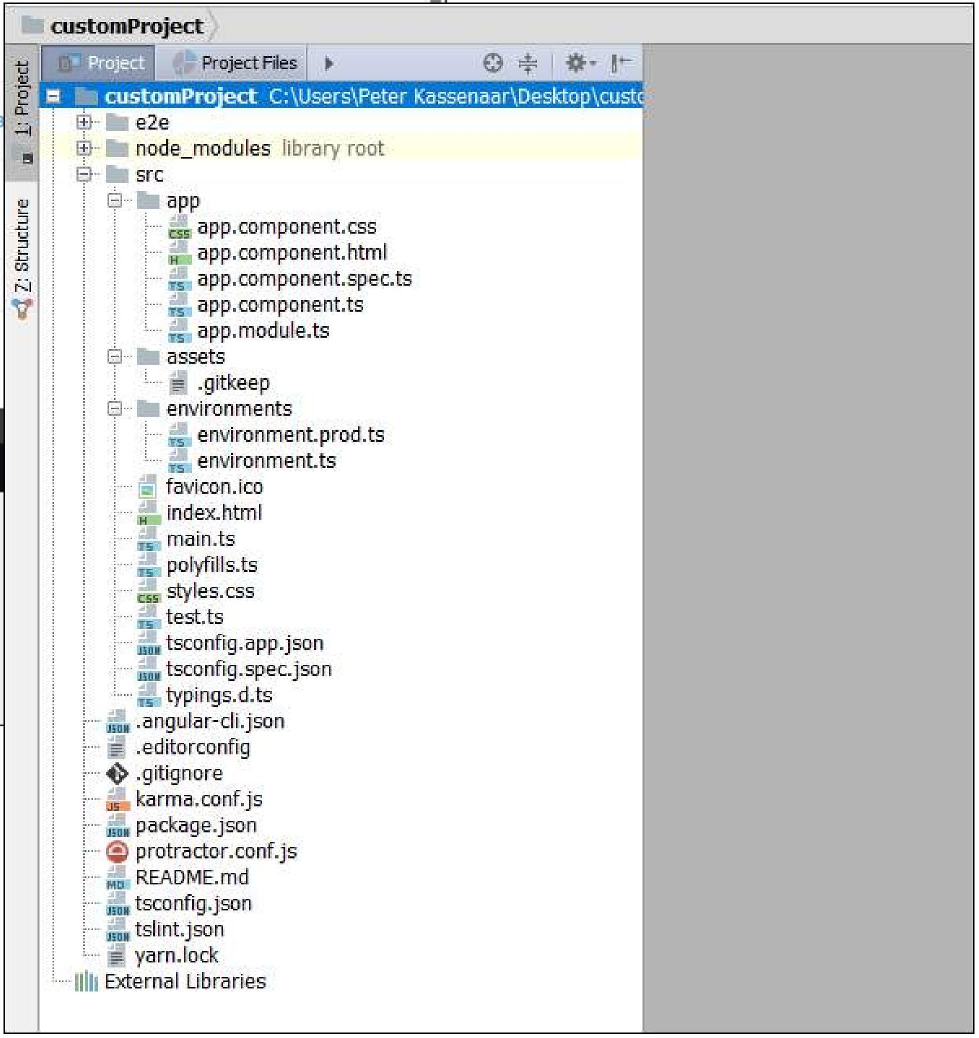
Multiple modules

Splitting your application into separate, reusable modules

Default application – 1 module


Welcome
multiple-modules app is running!
Resources
Here are some links to help you get started:
Learn Angular
CLI Documentation
Angular B
Next Steps
What do you want to do next with your app?
New Component
Angular Material
Add Dependency
Build for Production
Love Angular? Give our repo a star.
Star

```
ng generate component xyz
```



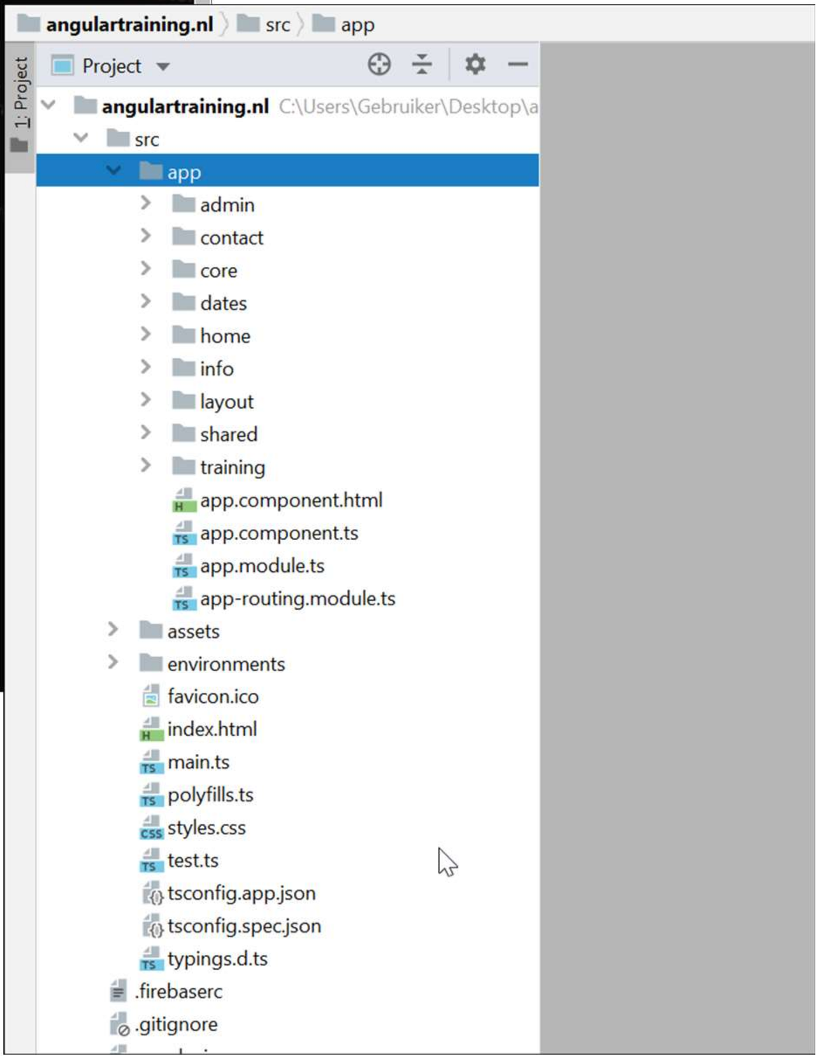
```
customProject
customProject  C:\Users\Peter Kassenaar\Desktop\custo
  e2e
  node_modules  library root
  src
    app
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
      .gitkeep
    environments
      environment.prod.ts
      environment.ts
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
    typings.d.ts
  .angular-cli.json
  .editorconfig
  .gitignore
  karma.conf.js
  package.json
  protractor.conf.js
  README.md
  tsconfig.json
  tslint.json
  yarn.lock
External Libraries
```

(228 MB)

Bigger applications – multiple modules

World-class Angular training in Dutch and English

Live classrooms - focused on today's developers

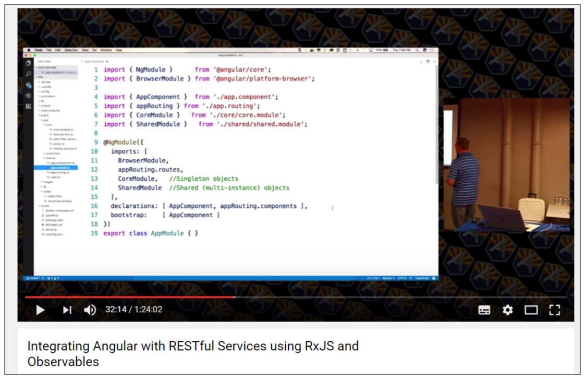


```
angulartraining.nl  C:\Users\Gebruiker\Desktop\a
  src
    app
      admin
      contact
      core
      dates
      home
      info
      layout
      shared
      training
      app.component.html
      app.component.ts
      app.module.ts
      app-routing.module.ts
    assets
    environments
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
    typings.d.ts
  .firebaserc
  .gitignore
```

Angular Modules

- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : code organization

- Use one `AppModule` - the root of your app
- Use one `CoreModule` - containing all *singletons* in your app
- Use one `SharedModule` - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- https://www.youtube.com/watch?v=YxK4UW4UfCk

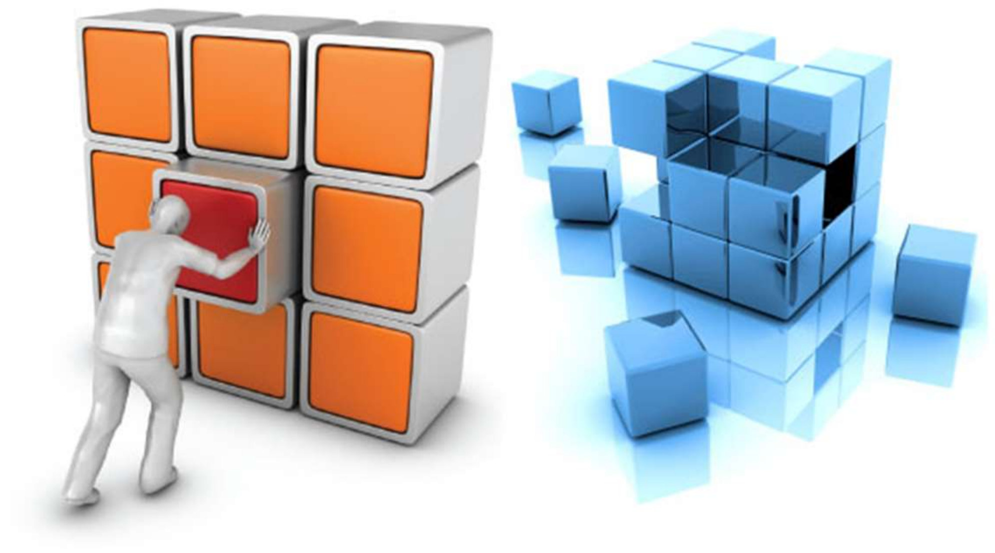


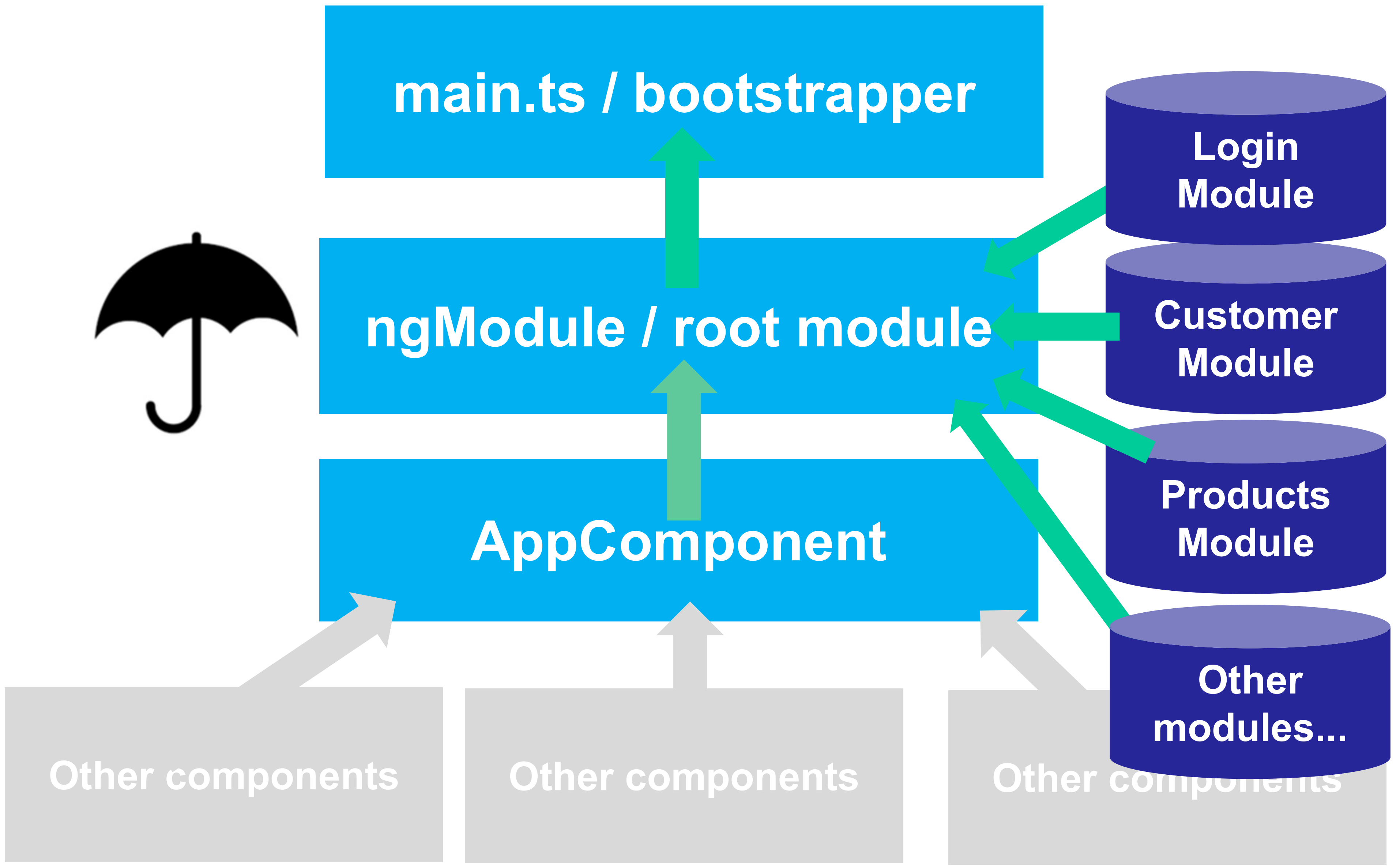
```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent }  from './app.component';
import { appRouting } from './app.routing';
import { CoreModule }   from './core/core.module';
import { SharedModule }   from './shared/shared.module';

@NgModule({
  imports: [
    BrowserModule,
    appRouting.routes,
    CoreModule,   //Singleton objects
    SharedModule  //Shared (multi-instance) objects
  ],
  declarations: [ AppComponent, appRouting.components ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Integrating Angular with RESTful Services using RxJS and Observables

Application – multiple Modules – why?

- *Reuse* of Components, Pipes, Routes and Services etc. over different apps
- *Wrap* each set of logical related components, services, etc. in its own module.




main.ts / bootstrapper
ngModule / root module
AppComponent
Login Module
Customer Module
Products Module
Other modules...
Other components
Other components
Other components

Steps

1. Create a new module
   - Optional: test first with `--dry-run`
   - `ng generate module customers --dry-run`
2. Create component(s) inside that module
   - Again: test first with `--dry-run`
   - `ng generate component customers --module customers --dry-run`
3. Apply UI, logic, etc. to your component
4. Export your component inside `customers.module.ts`
   - `exports : [CustomersComponent],`
   - Otherwise it can’t be used in other components!
5. Provide new module to `app.module.ts`
   - `imports: [CustomersModule]`

Optional : SharedModule

- Reuse components in multiple modules? Use a SharedModule
  - `ng g m shared` – shorthand notation

- Create components inside SharedModule

- Import SharedModule in other modules

- It doesn’t have to be in AppModule if you don’t use it directly!

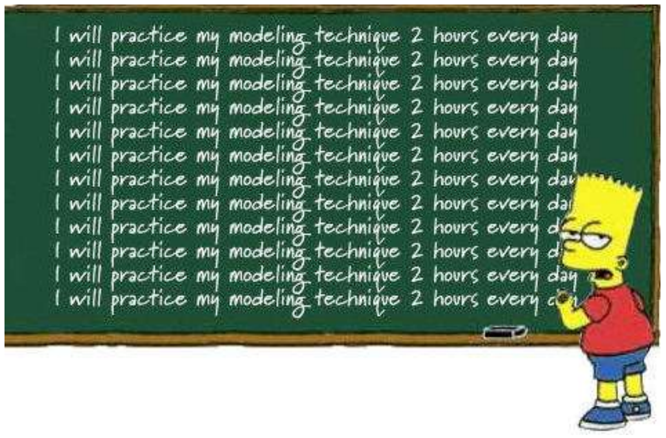
- It *does* not add size to module bundles

Demo

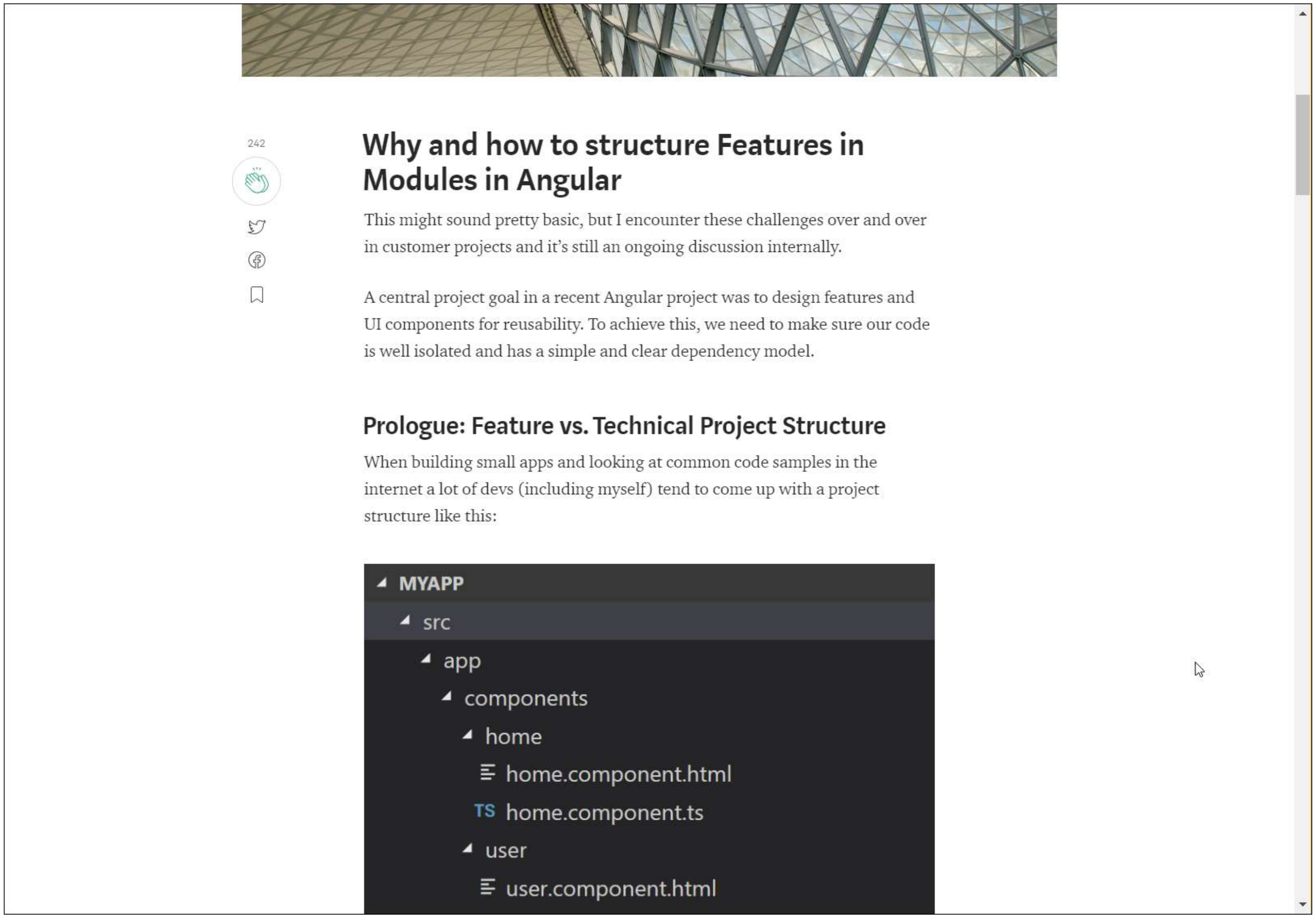
Workshop

- Open `../100-multiple modules`.
- Create a new module
- Create a new component inside this new module and give it some UI.
- Include the module in the Main Module and show it besides other modules
- Include the Search Component in your own module

- *OR:*
- Add Multiple Modules from scratch to your own application, using the steps described in this module.



How to structure feature modules


242

Why and how to structure Features in Modules in Angular

This might sound pretty basic, but I encounter these challenges over and over in customer projects and it's still an ongoing discussion internally.

A central project goal in a recent Angular project was to design features and UI components for reusability. To achieve this, we need to make sure our code is well isolated and has a simple and clear dependency model.

Prologue: Feature vs. Technical Project Structure

When building small apps and looking at common code samples in the internet a lot of devs (including myself) tend to come up with a project structure like this:

```
MYAPP
  src
    app
      components
        home
          home.component.html
          home.component.ts
        user
          user.component.html
```

https://medium.com/@philippbauknecht/why-and-how-to-structure-features-in-modules-in-angular-d5602c6436be